TDB-ACC-NO: NN9108115

DISCLOSURE TITLE: Tagged Inter Processor Communication Bus for Multiprocessor Systems.

PUBLICATION-DATA: IBM Technical Disclosure Bulletin, August 1991, US

VOLUME NUMBER: 34

ISSUE NUMBER: 3

PAGE NUMBER: 115 - 121

PUBLICATION-DATE: August 1, 1991 (19910801)

CROSS REFERENCE: 0018-8689-34-3-115

DISCLOSURE TEXT:

- A low-latency inter processor communication (IPC) bus structure is described that supports multiple processors communicating between each other and other devices on a shared bus. It utilizes a tagging scheme to uniquely identify multiple outstanding requests and supports simultaneous processor read requests and memory replies during the same bus cycle.

Introduction

One potential bottleneck in any bus-oriented multiprocessor system is the bandwidth required between processors and the shared memory system. The approach usually taken in designing the high bandwidth shared bus in a multiprocessor system is to provide wide data paths (e.g., 64 to 128 bits wide) and to use conventional uniprocessor bus protocols. That is, processors arbitrate for the bus and then keep it for the entire duration of the memory read or write cycle.

During this time, no other processor may communicate with another device on the shared bus. This results in a limited utilization of the bus, which accounts for one of the bottlenecks in bus-oriented multiprocessor systems.

- One way to minimize the time taken to perform read/write cycles on a multiprocessor shared bus is to decouple devices from the bus while the memory device is performing the actual read/write cycle. That is, processors simply need use the bus to transfer a read or write request, which memory devices consume. For the case of a write request, the memory device acknowledges the processor upon accepting the request and then proceeds to actually perform the memory write cycle. However, during this time, the shared bus can be used by processors to perform other shared memory requests. Similarly, for a read request, the memory device acknowledges the processor upon accepting the request, performs the memory read cycle, and then requests the bus to return the actual data. As before, the bus is free during the memory read cycle.
- Thus, assuming that the shared memory system consists of independently controlled banks of memory, several processor read/write requests can be serviced simultaneously. This significantly improves the bandwidth capability of the shared bus since, in practice, different processors will access distinct banks in the memory system. To accomplish this, processors need to "tag" memory read requests so that memory replies can be uniquely identified by the requesting processors. Each processor needs at least one unique tag but can use more to support several outstanding read requests.

IPC Bus

The IPC bus is a high-performance bus tailored to supporting multiple processors.

It is designed to maximize throughput for block- mode transfers, such as DMA transfers between disk controllers and memory, while minimizing average latency on single cycle shared memory references by processors. Furthermore, in order to minimize system complexity, and hence cost, a constant and narrow bus-width (32-bits of data plus 8-bits of error correction code (ECC)) is used throughout the system. The IPC bus provides a bandwidth 80 Mbytes/second (4- bytes every 50 ns bus cycle), which can be achieved by a single device performing a block-mode transfer, or by multiple devices performing back-to-back single cycle transfers.

- As shown in Fig. 1, the IPC bus is comprised of independent address and data buses (A-bus and D-bus). The A-bus consists of 30 address lines (full-word (4-bytes)) addressability is used, thus two additional lower-order address bits are assumed 0), 4 requester identifier lines (APID), 2 control lines (R/W and MODIFY), and a shared bus acknowledge line (BUSACK). This acknowledge line is used by address consumers, such as shared memory cards or processor IPC adapter (IPCA) cards, to signal requesters on the acceptance of a read/write bus cycle. The D-bus consists of 32 data lines, 8 ECC lines, and 4 data identifier lines (DPID). Since the A-bus and D-bus are decoupled, the APID and DPID lines are used to uniquely tag every read/write cycle and to support multiple outstanding requests from different sources.

- A central arbitration scheme is used to grant access to either the A-bus, D-bus or both. A processor wanting to perform an IPC bus read cycle issues an A-bus read request, drives its requesting address and unique requester identifier (APID) when granted the bus, and then waits for a reply on the D-bus. Replies to read requests return on a subsequent D-bus cycle, tagged with the requester's unique processor-id on the DPID lines. That is, during the corresponding D-bus cycle, the DPID will contain the unique processor ID placed on the APID by the requester when the read request was issued. Similarly, a processor wanting to perform an IPC bus write cycle issues an A-bus write cycle request to the central arbiter, and then drives the appropriate buses when granted.

Slave devices replying to read cycles request the D-bus for returning data, ECC, and data identifier (DPID).

- All devices attached to the IPC bus are synchronized to a common 20 MHz IPC clock, which defines an IPC bus cycle (50 ns/cycle). Arbitration is accomplished in a single 50 ns IPC clock cycle, and all single word transfers are performed in a single cycle. In addition, address and data cycles pertaining to a given processor are pipelined, which allows for overlapped processor read and memory reply cycles during the same IPC bus cycle. Several sample bus timing diagrams are shown in Figs. 2 and 3. Fig. 2 shows typical single-cycle IPC bus read and write transfers, while Fig. 3 shows typical burst-cycle timings.

IPC Bus Arbiter

As shown in Fig. 1, a central arbiter is used to grant access to the IPC bus.

The arbiter is comprised of two independent sections: one to service IPC bus master devices (e.g., processors and intelligent I/O devices), and another to service slave devices (e.g., shared memory, and display devices). The A-bus arbiter controls access to the A-bus and D-bus, while the D-bus arbiter controls access to the D-bus. Each master device is given a set of 3 independent control lines to gain bus access: a bus request line (R), a write request line (W), and a bus grant line (G). The write request line is used by processors to allocate the D-bus cycle immediately following the assigned A-bus cycle.

- The IPC bus arbiters use a round-robin arbitration scheme to grant access to the address and data buses. Once a given IPC bus device is granted access to the bus, it becomes the lowest priority device to be granted a subsequent cycle. Given that the D-bus is used by requesters wanting to perform IPC bus write cycles and responders wanting to return IPC bus read cycle replies, a simple interlock mechanism is used to arbitrate for the data bus. Write requests, which need both the A-bus and D-bus, are given priority over reply requests that only require the D-bus. Whenever a collision is detected between a write cycle and reply cycle, the D-bus arbiter holds off granting access to the memory device and queues up requests until the D-bus becomes available.

Processors wishing to perform an IPC bus write cycle assert their corresponding write request line (W) and issue an A-bus request (R). The A-bus arbiter, upon granting the bus A-bus to the processor (G), will lock out the D-bus arbiter from issuing a D-bus grant, thus granting the corresponding D-bus cycle to the processor.

- The A-bus arbiter provides a retry mechanism that automatically grants bus access to a requestor that failed to receive an acknowledgement (BUSACK) to a read/write bus cycle, without having to wait for its next prioritized turn. Failure to receive a BUSACK implies that the destination device is busy servicing a prior request. Typically, requesters issue a retry request once after failing to receive a BUSACK, and then issue normal bus requests to gain access to the IPC bus. A retry request is indicated by issuing an A-bus request and asserting the shared bus signal RETRY on the cycle following the bus acknowledge cycle pertaining to the previous requestor bus cycle. Fig. 4 illustrates a typical read and write cycle with retry requests. All requesters have a time-out mechanism for abnormal termination due to a non-existing address or device failure.
- The IPC bus handles both single cycle and burst mode transfers. As illustrated in Fig. 2, a device wishing to transfer a single address/data packet issues a one-cycle request pulse (AREQ) to the A-bus arbiter, which results in a one-cycle grant pulse (AGNT) after arbitration. The write request line (WREQ) is used to reserve the D-bus cycle immediately following the corresponding A-bus cycle for the case of a write operation. This line is asserted when the AREQ pulse is issued and de-asserted after receiving AGNT. Burst-mode transfers are accomplished by asserting the bus request line (AREQ) and holding it asserted until receiving the Nth-1 AGNT. For burst mode write transfers, WREQ is asserted along with AREQ and held until the Nth AGNT is received (see Fig. 3).
- During burst-mode transfers, the A-bus arbiter may issue the bus to another master device who needs bus access. Since the round-robin arbitration applies in both single-cycle and burst-mode transfers, the device performing a burst-mode transfer is always the lowest priority device after each bus cycle. Thus, multiple devices can be performing burst mode transfers in parallel, and the arbiter will give each device fair access to the IPC bus, while also allowing other devices to access the bus to perform single cycle transfers.
- The current IPC bus arbiter implementation supports 12 master devices and 12 slave devices, while the IPC bus and its attached devices support up to 15 outstanding read requests, and unlimited write requests (since a write request does not require a reply cycle). While the design can support far more devices and outstanding read requests, the numbers chosen are suitable for a small-scale parallel workstation with up to 12 processors and 12 memory cards.

United States. Contains confidential commercial information of IBM exempt from FOIA disclosure per 5 U.S.C. 552(b)(4) and protected under the Trade Secrets Act, 18 U.S.C. 1905.

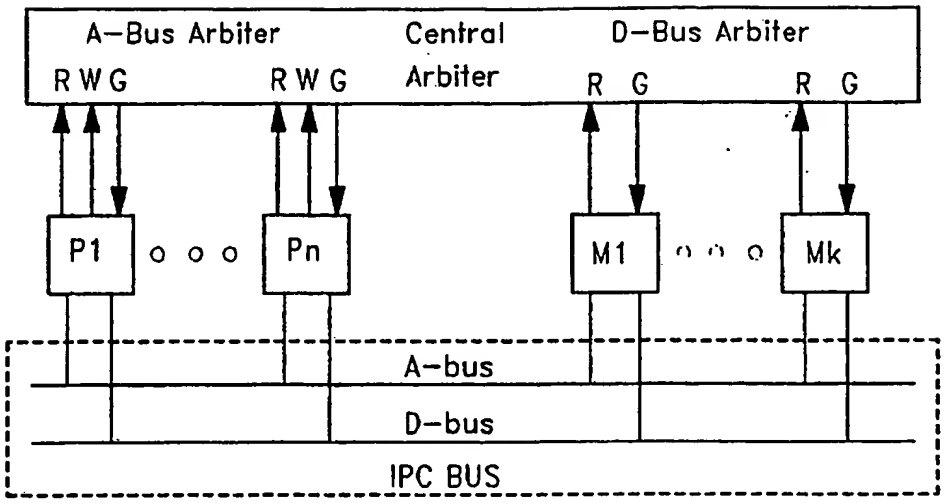


Figure 1. IPC Bus Structure

FIGURE 2A SINGLE-CYCLE IPC BUS READ TIMING

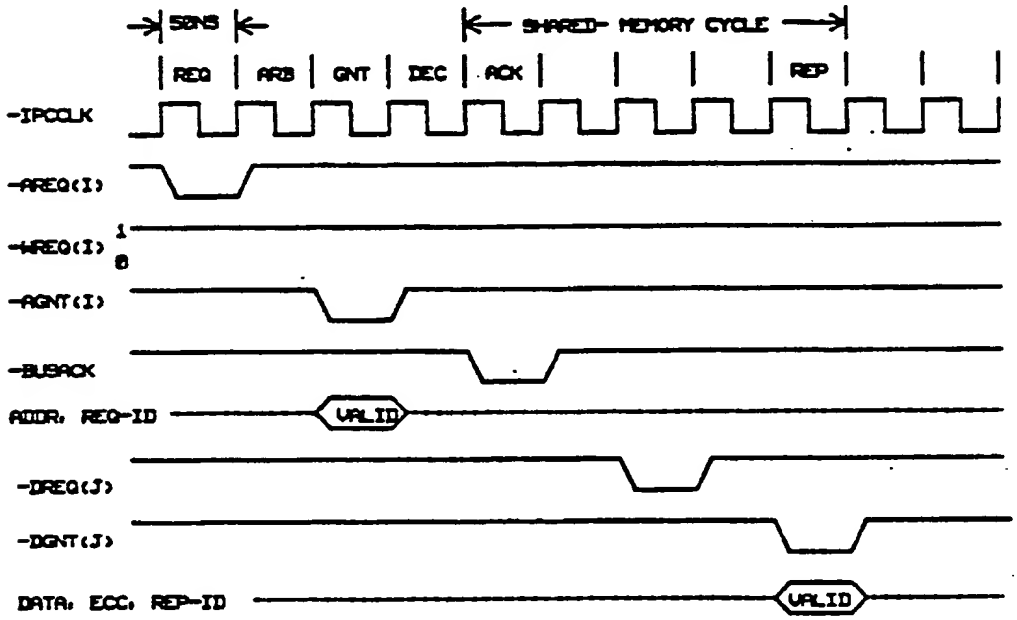


FIGURE 2B SINGLE-CYCLE IPC BUS WRITE TIMING

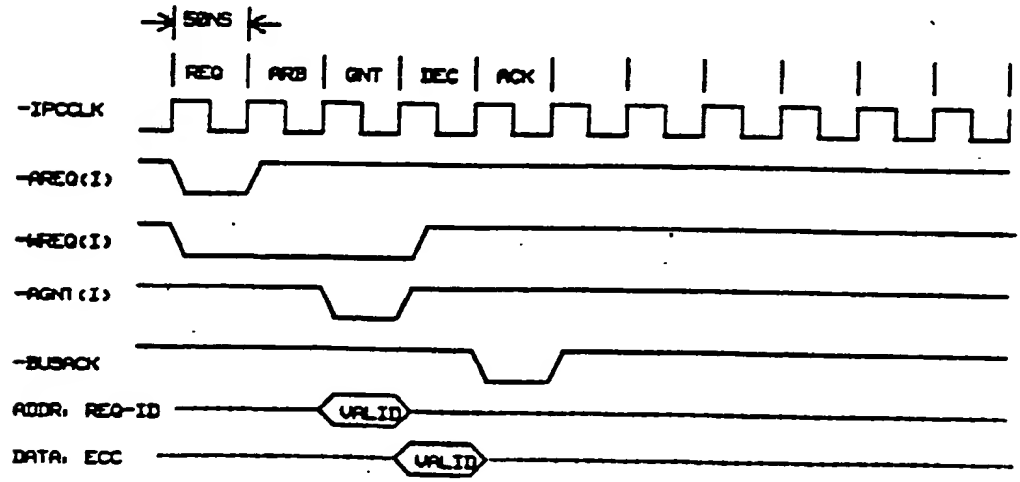


FIGURE 3A BURST-CYCLE IPC BUS READ TIMING

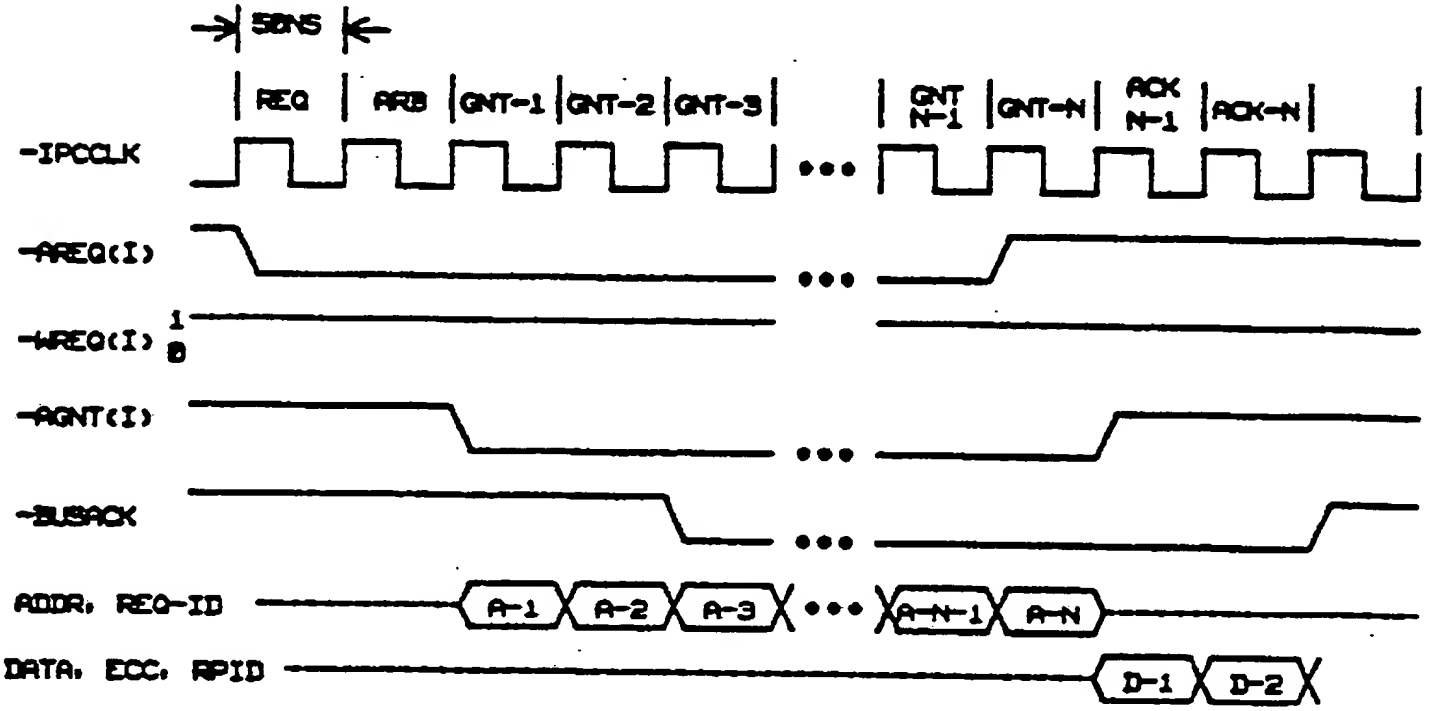


FIGURE 3B BURST-CYCLE IPC BUS WRITE TIMING

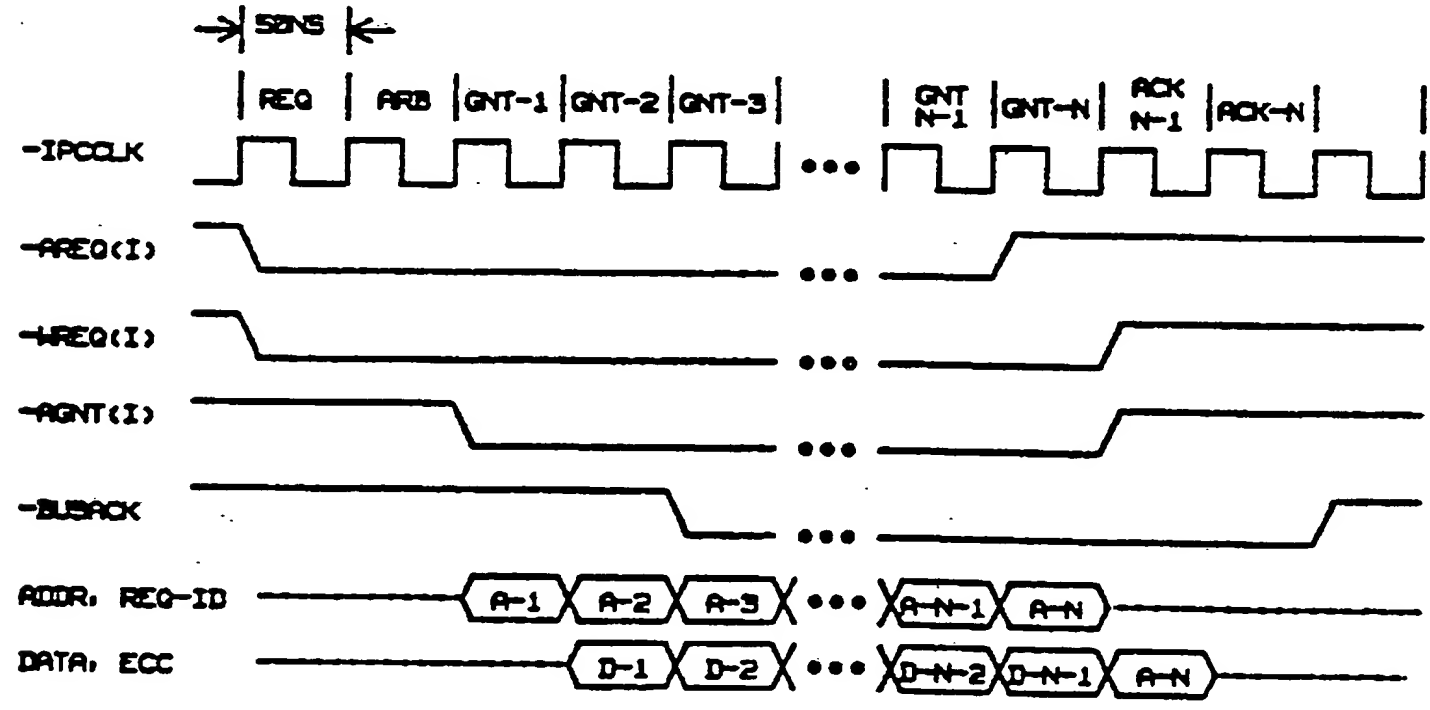


FIGURE 4A SINGLE-CYCLE IPC BUS READ TIMING WITH RETRY

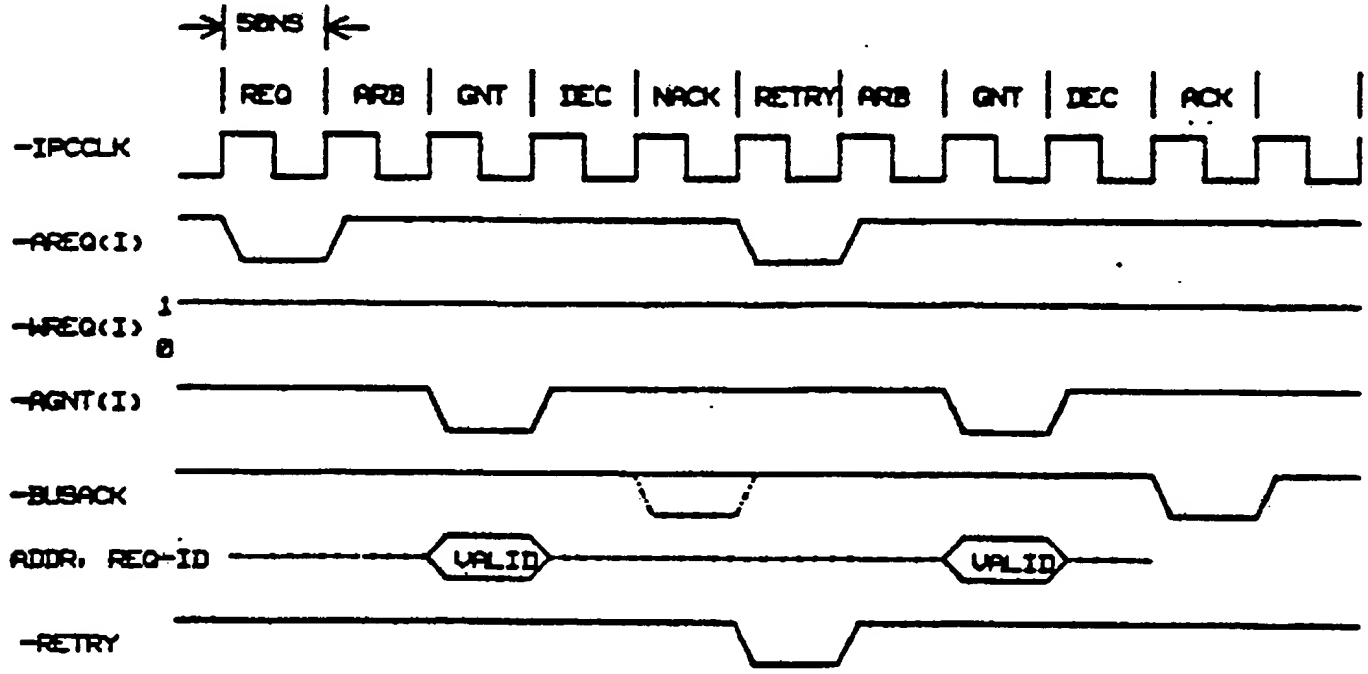
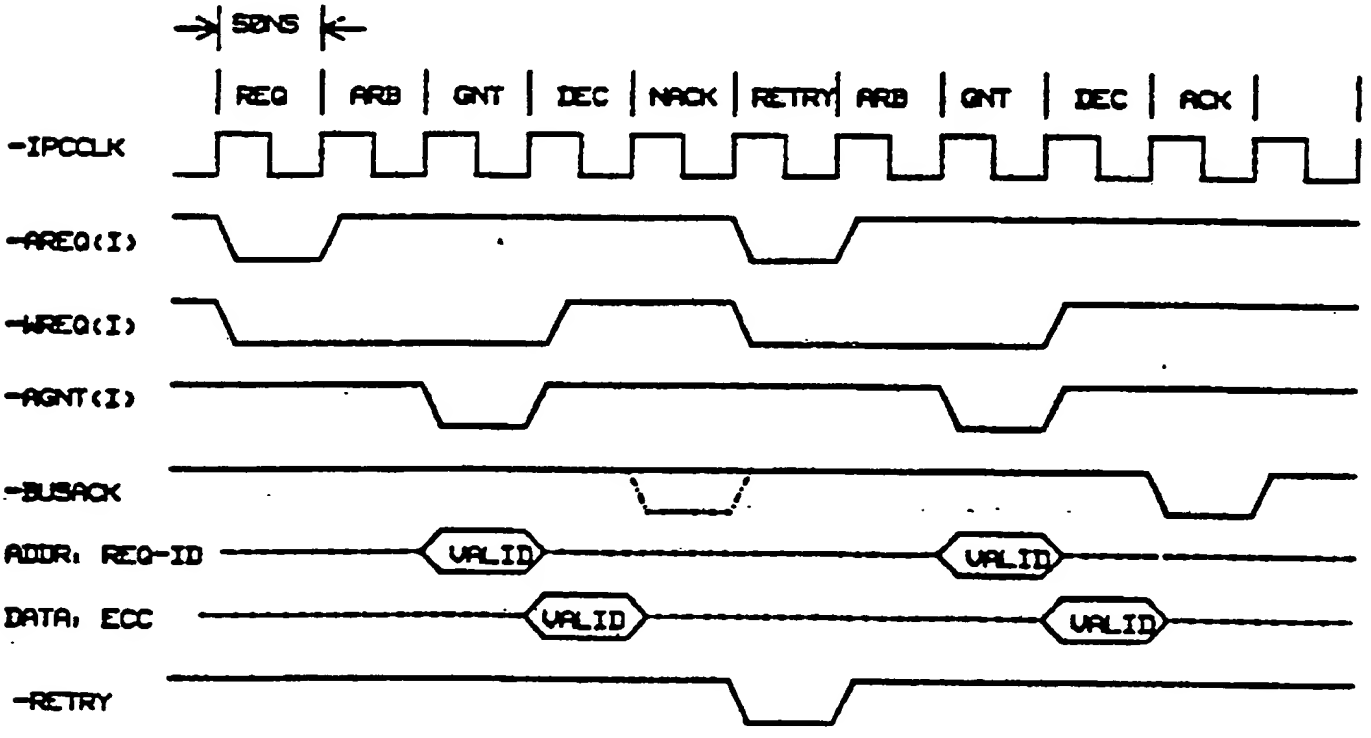


FIGURE 4B SINGLE-CYCLE IPC BUS WRITE TIMING WITH RETRY



P.121

L Number	Hits	Search Text	DB	Time stamp
1	10	route adj switch$3 adj packet$1 adj architecture$1	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/27 23:42
8	11144	request$1 near2 (bus or buses or busses)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/27 23:43
15	965	slave$1 same master$1 same (request$1 near2 (bus or buses or busses))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/27 23:51
22	139	arbiter$1 same grant$1 same (slave$1 same master$1 same (request$1 near2 (bus or buses or busses)))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/27 23:45
29	0	710/$.ccls. and 712/$.ccls. and 709/$.ccls. and (arbiter$1 same grant$1 same (slave$1 same master$1 same (request$1 near2 (bus or buses or busses))))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/27 23:45
36	3	710/$.ccls. and 709/$.ccls. and (arbiter$1 same grant$1 same (slave$1 same master$1 same (request$1 near2 (bus or buses or busses))))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/27 23:45
43	414	slave$1 near5 (request$1 near2 (bus or buses or busses))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/27 23:51
50	263	slave$1 near3 (request$1 near2 (bus or buses or busses))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/27 23:51
57	200	slave$1 near2 (request$1 near2 (bus or buses or busses))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/27 23:52
64	21	(arbiter$1 same grant$1 same (slave$1 same master$1 same (request$1 near2 (bus or buses or busses)))) same (slave$1 near2 (request$1 near2 (bus or buses or busses)))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/28 00:13
71	2	(split$1 near2 transaction$1) same (slave$1 near2 (request$1 near2 (bus or buses or busses)))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/28 00:15
78	8	(split$1 near2 transaction$1) same (slave$1 near5 (request$1 near2 (bus or buses or busses)))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/28 00:26
85	41218	(plural$4 or multi or multiple or two or dual) with protocol$1	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/28 00:27
92	17	(slave$1 same master$1 same (request$1 near2 (bus or buses or busses))) same ((plural$4 or multi or multiple or two or dual) with protocol$1)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/28 00:37

09/741,846

99	9	pipeline$1 same (arbiter$1 same grant$1 same (slave$1 same master$1 same (request$1 near2 (bus or buses or busses))))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2003/07/28 00:38